АНАЛИЗ МЕТРИК ПРОГРАММНОГО КОДА КАК СРЕДСТВА ПОВЫШЕНИЯ ПРОДУКТИВНОСТИ РАЗРАБОТКИ ПРОГРАММНОГО ПРОДУКТА

Кумаков В.В.
Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь
Новиков Е.В. – кандидат технич. наук, доцент

Представлен анализ метрик программного кода, приводится описание использования метрик, а также приводятся варианты использования метрик как средство повышения продуктивности разработки программного продукта.

Использование метрик программного кода позволяет разработчикам и руководителям проектов оценивать свойства создаваемого или уже существующего программного обеспечения(ПО), прогнозировать объем работ, давать количественную характеристику тех или иных проектных решений, качества разработанных систем и их частей, характеризовать сложность или надежность программного обеспечения. Также метрики можно применить и в оценке продуктивности разработки ПО, а, используя результаты измерений программного кода, можно скорректировать порядок работы программиста.

Так как подсчет и анализ метрик – процесс требующий определенного времени, метрики следует подсчитывать для уже существующей программы, когда выделяются основные метрики оценки кода, проводится анализ полученных метрик, либо после внесения изменений в программу с повторной оценкой ее метрических характеристик. Новые значения мер сравниваются как с эталонными значениями, так и с характеристиками программы до модификации. Таким образом можно выяснить сложность разработки внесенных изменений, а также эффективность разработчика.

При разработке программ и оценке усилий разработчиков различные метрики (особенно количественные) могут быть скомпрометированы со стороны не вполне добросовестных разработчиков, с целью как минимизировать, так и максимизировать те или иные используемые меры. В то же время, малая количественная оценка может свидетельствовать не о недостаточности усилий, а о сложности выполняемой работы – как разработки или модификации, так и анализа программ. Из этого следует сделать вывод, что для анализа программного кода следует использовать метрики различных видов, чтобы минимизировать недобросовестное или случайное влияние разработчика на оценку программного обеспечения.

Ряд количественных метрик, в основном используемых для оценки трудозатрат по проектам, основан на характеристиках исходного кода. Самой элементарной из таких метрик является SLOC (Source Lines Of Code – количество строк кода) Помимо количественных метрик также используются метрики сложности кода, которые включают в себя оценку количества структур в коде, связь между сложными структурами в коде(классы), использование объектов одних структур внутри других [1].

Для анализа сложности кода, существует список наиболее подходящих базовых метрик:

– количественные метрики: среднее число инструкций в функции, относительная сложность по Джилбу, ABC-метрика и обратная метрика размера базовых блоков. Данный тип метрик можно использовать для оптимизации объема кода с последующим уменьшением проекта и возможным ускорением работы кода;

– метрики сложности потока управления: цикломатическая сложность по Мак-Кейбу, метрика Хансена, метрика Харрисона и Мейджела, метрика Пивоварского, метрика граничных значений. С помощью данного типа метрик можно оценить связи внутри программы. Используя полученные данные, можно либо изменить, либо уменьшить связи внутри программы, что увеличит скорость понимания кода разработчиками;

– метрики сложности потока данных: метрика Чепина, спена, Кафура. Данный тип метрик используется для оценки использования переменных внутри программы. С помощью полученных данных можно регулировать использование переменных, что предотвратит чрезмерное появление переменных в программе [2].

Исходя из разновидности метрик и их анализа следует, что для повышения продуктивности разработки программного обеспечения следует обращать внимания на комплекс всех метрик, не выделяя при этом какие-то определенные метрики, так как они могут дать ложный результат и не оказать влияния на продуктивность работы.

Список использованных источников:

1. T.J. McCabe, "A complexity measure," IEEE Transactions on Software Engineering, vol. SE-2, no. 4, pp. 308-320, December, 1976

2. Arthur H.Watson, Thomas J. McCabe, “Structured Testing: A Testing Methodology Using Cyclomatic Complexity Metric”, NIST Special Publication 500-235, 1996